OpenMP

Miguel Alfonso Castro García
mcas@xanum.uam.mx

Universidad Autónoma Metropolitana - Izt

22 de junio de 2020

Contenido

Programación con memoria compartida

Herramienta de programación con MC mediante regiones paralelas

OpenMP (Open Multi-processing)

Herramienta para explotar el paralelismo en máquinas SMP o multiprocesador

- Soporte para paralelismo fino y grueso
- Librería para C, C++ y Fortran
- Provee paralelismo en loops
- Disminuye el uso directo de los mecanismos de sincronización

Programación con memoria compartida

Herramienta de programación con MC mediante regiones paralelas

OpenMP (Open Multi-processing)

Herramienta para explotar el paralelismo en máquinas SMP o multiprocesador

- Soporte para paralelismo fino y grueso
- Librería para C, C++ y Fortran
- Provee paralelismo en loops
- Disminuye el uso directo de los mecanismos de sincronización

Programación con memoria compartida

Herramienta de programación con MC mediante regiones paralelas

OpenMP (Open Multi-processing)

Herramienta para explotar el paralelismo en máquinas SMP o multiprocesador

- Soporte para paralelismo fino y grueso
- Librería para C, C++ y Fortran
- Provee paralelismo en loops
- Disminuye el uso directo de los mecanismos de sincronización

Programación con memoria compartida

Herramienta de programación con MC mediante regiones paralelas

OpenMP (Open Multi-processing)

Herramienta para explotar el paralelismo en máquinas SMP o multiprocesador

- Soporte para paralelismo fino y grueso
- Librería para C, C++ y Fortran
- Provee paralelismo en loops
- Disminuye el uso directo de los mecanismos de sincronización

Programación con memoria compartida

Herramienta de programación con MC mediante regiones paralelas

OpenMP (Open Multi-processing)

Herramienta para explotar el paralelismo en máquinas SMP o multiprocesador

- Soporte para paralelismo fino y grueso
- Librería para C, C++ y Fortran
- Provee paralelismo en loops
- Disminuye el uso directo de los mecanismos de sincronización

Programación con memoria compartida

Herramienta de programación con MC mediante regiones paralelas

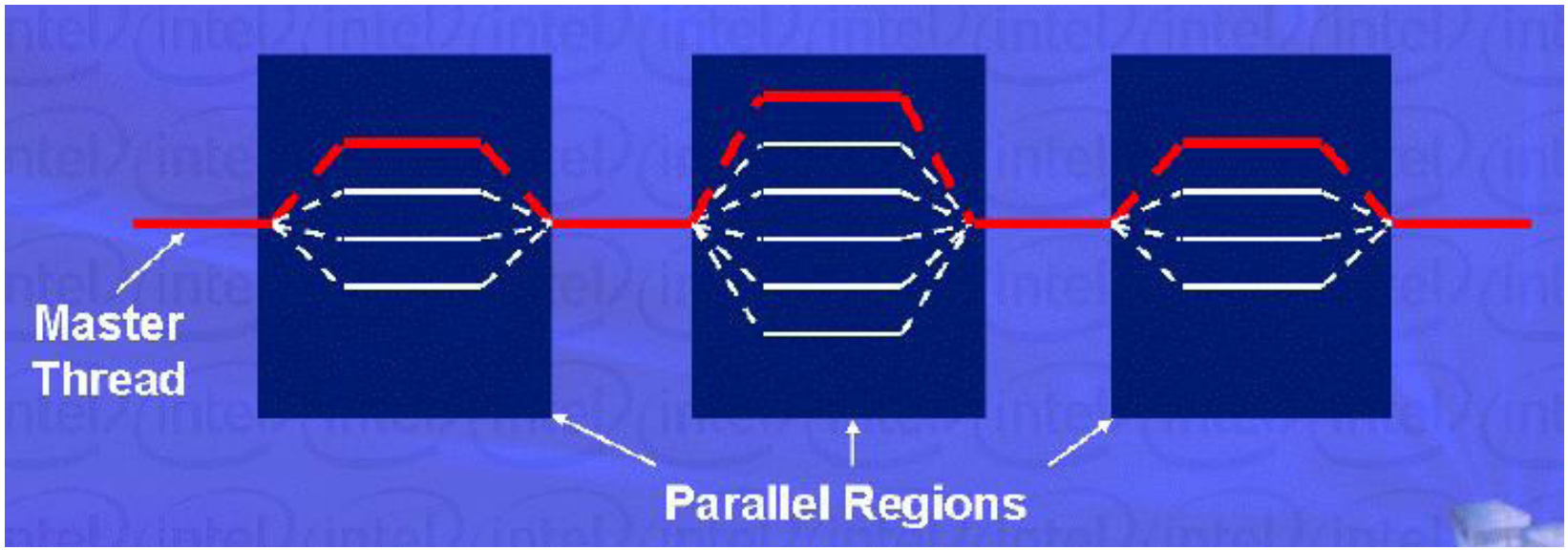
OpenMP (Open Multi-processing)

Herramienta para explotar el paralelismo en máquinas SMP o multiprocesador

- Soporte para paralelismo fino y grueso
- Librería para C, C++ y Fortran
- Provee paralelismo en loops
- Disminuye el uso directo de los mecanismos de sincronización

OpenMP

El paralelismo (fork-join) se da mediante la ejecución de una tarea, compuesta de una secuencia de pasos secuenciales combinados con regiones paralelas



OpenMP

Las directivas de OpenMP en un programa en C se indican mediante

```
#pragma omp directivaOMP [parámetros]
```

Ejemplos

```
#pragma omp parallel
#pragma omp for
#pragma omp section
#pragma omp critical
#pragma omp barrier
```

OpenMP

Las directivas de OpenMP en un programa en C se indican mediante

```
#pragma omp directivaOMP [parámetros]
```

Ejemplos

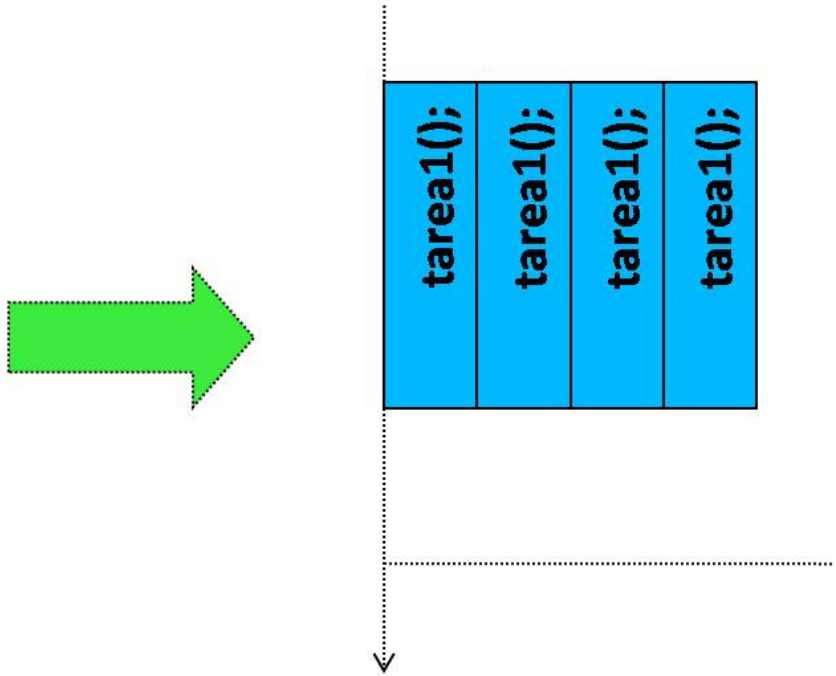
```
#pragma omp parallel
#pragma omp for
#pragma omp section
#pragma omp critical
#pragma omp barrier
```

OpenMP: parallel

```
#pragma omp parallel
{
  tarea1();
}
```

OpenMP: parallel

```
#pragma omp parallel
{
  tarea1();
}
```

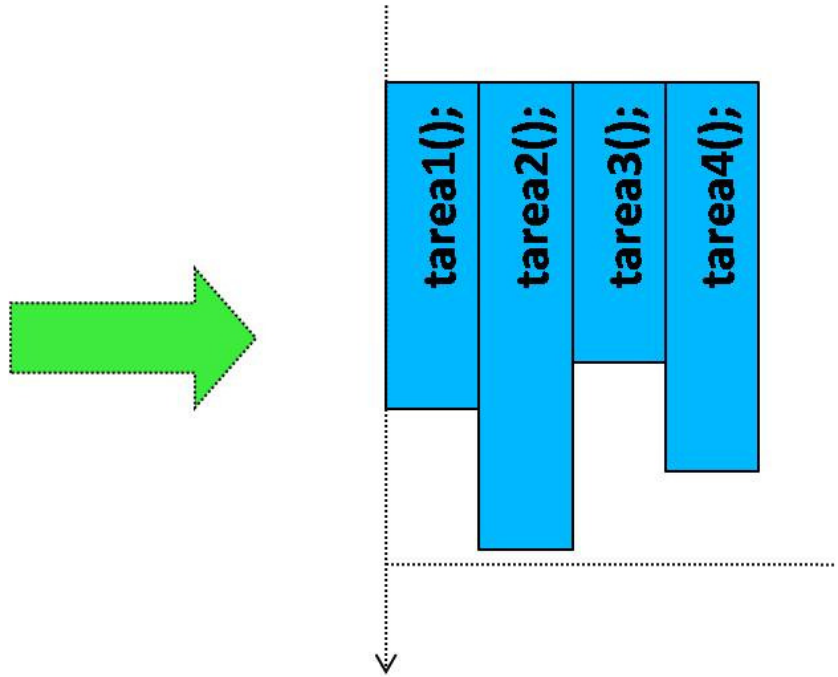


OpenMP: parallel sections

```
#pragma omp parallel sections
{
  # pragma omp section
  tarea1();
  # pragma omp section
  tarea2();
  # pragma omp section
  tarea3();
  # pragma omp section
  tarea4();
}
```

OpenMP: parallel sections

```
#pragma omp parallel sections
{
  # pragma omp section
  tarea1();
  # pragma omp section
  tarea2();
  # pragma omp section
  tarea3();
  # pragma omp section
  tarea4();
}
```



Funciones

Algunas rutinas especiales de OpenMP

- **omp_set_num_threads(int);**
  Se define el número de hilos a usar
- **omp_get_thread_num(void);**
  Se obtiene el identificador del hilo (entero >= 0)
- **omp_get_num_threads(void);**
  Se obtiene el número total de hilos
- **omp_get_num_procs(void);**
  Se obtiene el número de procesadores utilizado

Funciones

Algunas rutinas especiales de OpenMP

- **omp_set_num_threads(int);**
  Se define el número de hilos a usar
- **omp_get_thread_num(void);**
  Se obtiene el identificador del hilo (entero >= 0)
- **omp_get_num_threads(void);**
  Se obtiene el número total de hilos
- **omp_get_num_procs(void);**
  Se obtiene el número de procesadores utilizado

Funciones

Algunas rutinas especiales de OpenMP

- **omp_set_num_threads(int);**
  Se define el número de hilos a usar
- **omp_get_thread_num(void);**
  Se obtiene el identificador del hilo (entero >= 0)
- **omp_get_num_threads(void);**
  Se obtiene el número total de hilos
- **omp_get_num_procs(void);**
  Se obtiene el número de procesadores utilizado

Funciones

Algunas rutinas especiales de OpenMP

- **omp_set_num_threads(int);**
  Se define el número de hilos a usar
- **omp_get_thread_num(void);**
  Se obtiene el identificador del hilo (entero $>= 0$)
- **omp_get_num_threads(void);**
  Se obtiene el número total de hilos
- **omp_get_num_procs(void);**
  Se obtiene el número de procesadores utilizado

Funciones

Algunas rutinas especiales de OpenMP

- **omp_set_num_threads(int);**
  Se define el número de hilos a usar
- **omp_get_thread_num(void);**
  Se obtiene el identificador del hilo (entero >= 0)
- **omp_get_num_threads(void);**
  Se obtiene el número total de hilos
- **omp_get_num_procs(void);**
  Se obtiene el número de procesadores utilizado

Rutinas

- **omp_init_lock(omp_lock_t*);**
  Inicialización de un candado
- **omp_set_lock(omp_lock_t*);**
  Operación para cerrar un candado
- **omp_unset_lock(omp_lock_t*);**
  Operación para abrir un candado
- **omp_test_lock(omp_lock_t*);**
  Devuelve V si el candado no está cerrado
- **omp_destroy_lock(omp_lock_t*);**
  Destrucción de un candado

Rutinas

- **omp_init_lock(omp_lock_t*);**
  Inicialización de un candado
- **omp_set_lock(omp_lock_t*);**
  Operación para cerrar un candado
- **omp_unset_lock(omp_lock_t*);**
  Operación para abrir un candado
- **omp_test_lock(omp_lock_t*);**
  Devuelve V si el candado no está cerrado
- **omp_destroy_lock(omp_lock_t*);**
  Destrucción de un candado

Rutinas

- **omp_init_lock(omp_lock_t*);**
  Inicialización de un candado
- **omp_set_lock(omp_lock_t*);**
  Operación para cerrar un candado
- **omp_unset_lock(omp_lock_t*);**
  Operación para abrir un candado
- **omp_test_lock(omp_lock_t*);**
  Devuelve V si el candado no está cerrado
- **omp_destroy_lock(omp_lock_t*);**
  Destrucción de un candado

Rutinas

- **omp_init_lock(omp_lock_t*);**
  Inicialización de un candado
- **omp_set_lock(omp_lock_t*);**
  Operación para cerrar un candado
- **omp_unset_lock(omp_lock_t*);**
  Operación para abrir un candado
- **omp_test_lock(omp_lock_t*);**
  Devuelve V si el candado no está cerrado
- **omp_destroy_lock(omp_lock_t*);**
  Destrucción de un candado

Rutinas

- **omp_init_lock(omp_lock_t*);**
  Inicialización de un candado
- **omp_set_lock(omp_lock_t*);**
  Operación para cerrar un candado
- **omp_unset_lock(omp_lock_t*);**
  Operación para abrir un candado
- **omp_test_lock(omp_lock_t*);**
  Devuelve V si el candado no está cerrado
- **omp_destroy_lock(omp_lock_t*);**
  Destrucción de un candado

Hola Mundo

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
  #pragma omp parallel
    printf("Hello world \n");
  return 0;
}
```

Tids

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{ int nthreads,tid;

  #pragma omp parallel private (nthreads,tid)
  { tid = omp_get_thread_num(); //obtiene número de hilo
    printf("Soy el hilo %d",tid);
    if(tid == 0)
    { nthreads = omp_get_num_threads();
      printf("Número total de hilos %d",nthreads);
    }
  }
printf("Fin del programa");
}
```

Suma paralela

```
#include <omp.h>
void main()
{ int A[256], suma_total=0;
  Ini_arreglo(A);

  omp_set_num_threads(omp_num_procs());
  #pragma omp parallel shared (A,suma_total)
  { int ini,fin,suma=0;
    int id = omp_get_thread_num();
    LimitesArr(id, 256, omp_num_procs(),&ini, &fin);
    for(i=ini; i<=fin; i++)
        suma=suma+A[i];
    #pragma omp critical
    {
      suma_total+=suma;
    }
  }
  printf("Suma total:%d ",suma_total);
}
```